

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



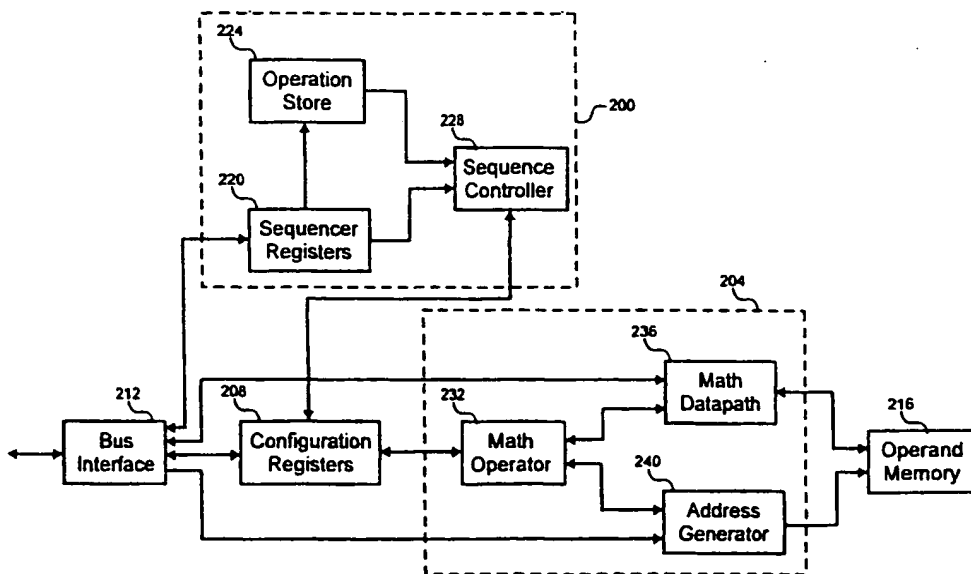
(43) International Publication Date
14 December 2000 (14.12.2000)

PCT

(10) International Publication Number
WO 00/76119 A1

- (51) International Patent Classification⁷: H04L 9/32
- (21) International Application Number: PCT/US00/15872
- (22) International Filing Date: 8 June 2000 (08.06.2000)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
60/138,147 8 June 1999 (08.06.1999) US
09/393,147 10 September 1999 (10.09.1999) US
- (71) Applicant (for all designated States except US): GENERAL INSTRUMENT CORPORATION [US/US]; 101 Tournament Drive, Horsham, PA 19044 (US).
- (72) Inventor; and
- (75) Inventor/Applicant (for US only): SIMON, Daniel, Z. [US/US]; 11135 Affinity Court #19, San Diego, CA 92131 (US).
- (74) Agents: KULAS, Charles, J. et al.; Townsend and Townsend and Crew LLP, Two Embarcadero Center, Eighth Floor, San Francisco, CA 94111 (US).
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).
- Published:
— With international search report.
- For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: CRYPTOGRAPHIC PROCESSING SYSTEM



(57) Abstract: Methods and apparatuses which allow for execution of a number encryption functions by a specialized encryption processor (108, 200, 204, 208, 212, 216) are disclosed. In one embodiment, a method processes cryptographic functions. A function is received which is comprised of a number of commands. The commands include at least a loop or branch. At least one of the commands is converted to a number of subcommands. After conversion, the number of subcommands are executed.

CRYPTOGRAPHIC PROCESSING SYSTEM

This invention related in general to cryptographic processing systems and more specifically to apparatuses and methods for allowing execution of a number
5 encryption functions by a specialized encryption processor.

BACKGROUND OF THE INVENTION

Cryptographic processing systems are used to provide security to data transmissions. Implementation of the cryptographic algorithms can introduce latencies
10 which are undesirable in data transmission. Accordingly, there is a general need to improve the data transmission speed through cryptographic systems.

Some conventional systems use general purpose processors and software to perform cryptographic processing. The algorithms reside in firmware and are executed like any other software. The firmware implements the algorithms with the instruction set
15 of the general purpose processor. If the algorithms change, these systems can be reprogrammed with new firmware to implement the new algorithms. The general purpose processors perform the cryptographic computations slowly because the instruction set is not specialized to include cryptographic instructions. For example, a general purpose processor may only have multiply instruction where some cryptographic
20 algorithms routinely use Montgomery multiplications. Accordingly, performing a Montgomery multiplication would require execution of a series of processor instructions on a general purpose processor. For this reason, general purpose processors execute cryptographic algorithms slowly which increases data transmission latency.

Additionally, general purpose processors have functionality not required in
25 cryptographic processors. For example, some general purpose processors include branch prediction, integrated interfaces and interrupt support. These processors have evolved to provide a moderate level of support to many different functions. As a result, the specialized needs of cryptography are not well served because of the overhead needed to support other functions. For example, servicing an interrupt while calculating a
30 Montgomery multiply will slow down the execution of that algorithm which may increase latency. Accordingly, general purpose processors are slowed down by the overhead required to process different types of tasks.

Conventional systems have attempted to solve these problems by relying upon specialized cryptographic processors. For example, specialized circuits which implement DES algorithms are well known. These systems have specialized hardware which quickly implements a desired cryptographic algorithm such as encrypt or decrypt.

5 These specialized circuits are typically very large in order to quickly implement the desired function. In contrast to the hardware in the general purpose processor, little of the circuitry is reused in a conventional cryptographic processor.

Unlike the software implementations, the specialized cryptographic processors cannot be reprogrammed. Once the hardware for the algorithms is produced,

10 it cannot be changed. Accordingly, changing the algorithm would require redesigning the circuit and replacing all integrated circuits incorporating the algorithm. The ability to change the algorithm is desirable because it produces a moving target for hackers or pirates which may attempt to circumvent the cryptographic process.

In summary, it appears desirable to develop a cryptographic processor

15 which is faster than software implementations run on general purpose processors. Further, the implementation should be flexible to allow for changes in the algorithm and should require less circuitry to implement than specialized hardware cryptographic processors.

20 SUMMARY OF THE INVENTION

According to the invention, apparatuses and methods allow for execution of a number encryption functions by a specialized encryption processor. In a first embodiment, a method for executing a plurality of commands in a cryptographic processing system is disclosed. A first set of commands are received and executed,

25 whereafter a flag is set. A second set of commands are also received and executed.

In another embodiment, a cryptographic processing system includes a cryptographic processor, a general purpose processor and a bus. The cryptographic processor executes a function which is sent by the general purpose processor. For this purpose, the bus couples the cryptographic processor to the general purpose processor.

30 Another embodiment describes a cryptographic processor which includes a bus interface, a sequencer, operand memory, and a math unit. The sequencer stores a function received from the bus interface. The math unit is coupled to the bus interface, operand memory and sequencer.

In yet another embodiment, a method for processing cryptographic functions is disclosed. A function is received which is comprised of a number of commands. The commands include at least a loop or branch. At least one of the commands is converted to a number of subcommands. After conversion, the number of
5 subcommands are executed.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram depicting one embodiment of the cryptographic processing system;

10 Fig. 2 is a block diagram which illustrates an embodiment of a cryptographic engine;

Fig. 3 is a flow diagram which shows a process for performing a single datapath operation under the control of a central processing unit; and

15 Fig. 4 is a flow diagram which shows a process for performing a cryptographic function comprised of a number of datapath operations under the control of a sequencer.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

While this invention is susceptible of embodiments in many different
20 forms, there is shown in the drawings and will herein be described in detail, a number of embodiments of the invention with the understanding that the present disclosure is to be considered as an exemplification of the principles of the invention and is not intended to limit the broad aspects of the invention to the embodiment illustrated. In the Figures, similar components and/or features may have the same reference label.

25 A block diagram of one embodiment of the cryptographic processing system 100 is illustrated in Fig. 1. The system 100 includes a central processing unit (CPU) 104, a crypto engine 108, one or more peripherals 112, memory 116, and firmware storage memory 120 which are all interconnected by a system bus 124. Because the CPU 104 is inefficient at performing cryptographic computations, they are passed over the
30 system bus 124 to the crypto engine 108 for execution. Either a single datapath operation or a complete function containing multiple datapath operations may be passed to the crypto engine 108 for execution.

The CPU 104 generally controls operation of the cryptographic processing system 100. The CPU 104 is a general purpose processor which performs a variety of

tasks. To support this variety, general purpose processors 104 typically have many more instructions than, for example, the cryptographic engine 108. When cryptographic processing is required, the CPU 104 has the cryptographic engine perform the processing. In the mean time, the CPU 104 can process other tasks needed to control the processing system 100. Preferably, the CPU 104 is a MIPS[®] embedded core, however, any number of processing cores could be used.

The CPU 104 uses software code or firmware to control operation of the cryptographic processing system 100. In this embodiment, the code is stored in firmware storage memory 120 which is a nonvolatile memory such as battery backed random access memory (RAM).

The CPU 104 uses additional memory 116 to assist execution of the code within the CPU 104. This memory 116 is volatile and rewritable memory which does not retain its value if power is removed. However, other embodiments could use nonvolatile and rewritable form of memory. Variables and data needed during program execution are stored in this memory 116.

A number of peripherals 112 are used by the CPU 104 to perform specialized tasks. For example, in the context of a television set-top box, these peripherals 112 could be dedicated to such tasks as modem data transmission to a telephone network. The information sent to and received from the telephone network would be converted by the peripheral 112 into a format used by the CPU 104 such that the CPU 104 could interface with the telephone network using the peripheral 112.

Also attached to the system bus 124 is the crypto engine 108 which performs any cryptographic algorithms. The crypto engine 108 has limited functionality specialized for efficient cryptography. In contrast, the CPU 104 has greater functionality but cannot perform cryptographic functions efficiently. Preferably, the crypto engine 108 performs computations required for the Rivest, Shamir and Adleman (RSA) cryptographic algorithm. However, other embodiments could perform computations associated with any symmetric or asymmetric cryptographic algorithm. Although not shown, interrupts are used as flags to indicate to the CPU 104 when processing by the crypto engine 108 is complete. However, other embodiments could use polled interrupts or any other technique for setting a flag.

The system bus 124 allows communication between all circuits attached to the bus. The system bus 124 allows sending information between the CPU 104 and any other circuit attached to the system bus 124. Additionally, a block of information can be

exchanged between the memory 116 and the crypto engine 108 using a direct memory access (DMA) circuit. To perform a DMA transfer, the CPU 104 places a number of data words in the system memory and tells a DMA circuit the output for the data. The DMA transfer directly transfers the data from memory to the crypto engine 108 over the system
5 bus 124. As can be appreciated, DMA circuits can considerably improve the bandwidth of data transfer because the latencies of the CPU 104 shepherding the transfer of each word of data are avoided.

Referring next to Fig. 2, an embodiment of the crypto engine 108 is shown in block diagram form. The crypto engine 108 includes a sequencer 200, a math unit 204,
10 configuration registers 208, a bus interface 212, and operand memory 216. This embodiment can accept from the CPU 104 either a single datapath operation or a function including a number of datapath operations. Preferably, the transfer of data associated with executing a function or datapath operations uses DMA transfers.

The bus interface 212 attaches to the system bus 124 and allows data
15 exchange with the crypto engine 108. The bus interface 212 contains control logic which grafts handshaking and data transfer needs of the crypto engine 108 into the memory space of the CPU 104. From the viewpoint of the system bus 124, all the interaction with the crypto engine 108 is memory mapped into the address space of the CPU 104. Some of the circuitry required to enable DMA transfers is also located in the bus interface 212,
20 but most of the DMA circuitry is located elsewhere in the cryptographic processing system 100.

The bus interface 212 passes information between the CPU 104 and the configuration registers 208, sequencer 200 and math unit 204. The configuration registers 208 contain parameters associated with execution of a datapath operation such as operand
25 length, datapath operation desired, start address of operand, etc. To perform a datapath operation, the math unit 204 uses the information in the configuration registers written by the CPU 104 or sequencer 200.

The math unit 204 performs datapath operations upon the operands in the operand memory. Which operation to perform and which operands to perform the
30 datapath operation upon are designated in the configuration registers 208. Once the information required to execute an datapath operation is written into the configuration registers 208, the math unit 208 waits for the writing into the configuration registers 208 of a flag which indicates processing should commence. Next, the math unit 208 reads the

input operands and configuration information and executes the datapath operation. After execution, a resulting output operand is written into the operand memory 216.

The math unit 204 includes a number of submodules such as a math operator 232, an address generator 240 and a math datapath 236. The math operator 232
5 retrieves the datapath operation from the configuration registers 208 and manages execution of that datapath operation. A state machine within the math operator 232 determines the operation desired, controls the math datapath 236 to perform the arithmetic and controls the address generator 240 to load and store operands from the operand memory 216.

10 The math datapath 236 performs multiplies, squares, additions, subtractions, shifts, Montgomery multiplies, and various other operations on words of data. The operands are comprised of a number of words which are subdivided down to single words during execution by the math operator 232 prior to processing by the math datapath 236. For example, the input operands are 1024 bits, but the word size processed
15 by the math datapath 236 is thirty-two bits. The math operator 232 iteratively has the math datapath perform thirty-two bit word operations to achieve a desired 1024 bit datapath operation.

There are standard algorithms which allow performing arithmetic on large operands by using smaller arithmetic functions. However, these standard algorithms rely
20 upon software which executes much slower than hardware, unlike the math operator 232 which controls the process with hardware state machines. For example, a 1024 bit addition can be achieved by sequentially adding corresponding words of both input operands and including the carry from any previous word addition. To add two 1024 bit numbers, thirty-two consecutive additions with carry of thirty-two bit words are required.
25 Known algorithms similarly exist for subtractions, multiplies, squares, bit shifts, and other arithmetic functions.

Since the math datapath 236 only performs thirty-two bit word operations, the complexity and size of that circuit is greatly reduced. As can be appreciated, a 1024 bit multiply is an extremely large circuit if the multiply is not subdivided into word
30 operations. Accordingly, the present invention reduces circuit size and complexity without resorting to the use of software algorithms performed on a general purpose processor.

The address generator 240 and math datapath 236 work under the control of the math operator 232 to perform the word operations. The state machine in the math

operator 232 causes the address generator 240 to activate the proper connection between the math datapath 236 and operand memory 216 when reading and writing operands. Additionally, the state machine connects the input operand word and output operand word to the correct word operation in the math datapath 236. For example, in the case of an
5 addition word operation, the state machine connects the appropriate words of the input operands from the operand memory 216 by manipulating the address generator 240 to the input of the addition operation. The output word from the addition is written to the output operand word addressed by the address generator 240. Any carry out from the addition is retained by the math datapath 236 as the carry in for the next word addition.

10 The operand memory 216 stores the input written by the CPU 104 and output operands written by the math datapath 236. Additionally, intermediate results from executing the datapath operation may also be stored in the operand memory 216. In this embodiment, the operand memory 216 is organized in eight blocks, where each block is 2048 bits long. Each operand can span up to two blocks and have a size of up to 4096
15 bits. Each block is subdivided into sixty-four words which are each thirty-two bits wide. When the CPU writes the input operands, the configuration registers 208 are also written with the length of the each input operand, the block(s) which contain each operand and the start address of the operand within those block(s). In this way, an operand may be written into any location of the operand memory and need not be aligned with any block
20 boundary. In a similar way, the location of the output operand is designated in the configuration registers 208.

The operand memory 216 is designed for efficiently writing and reading operands. Preferably, dual port memory is used as the operand memory 216. There are three read buses and one is a write bus data attached to the operand memory 216. The
25 three read busses share one port on each memory cell while the write bus has its own port on each memory cell. However, only one bus is able to read or write to a given block at a time so the datapath algorithms are designed such that they do not attempt to simultaneously access the two ports of the memory cell at the same time. Since there are three read buses and one write bus, there can be three input operands and one output
30 operand for a given word operation. Addressing of the operand memory 216 includes a select bus which chooses the proper block and a word address bus which selects the proper word within that block. There are additional enable lines and bus gating lines to enable the above described functionality.

The CPU 104 can either pass a single datapath operation to the crypto engine 108 for execution or can send a function which contains a number of datapath operations for execution. An example of a function would be some type of RSA task such as key generation. In the case where a function is sent for execution, the sequencer 200 is employed to execute the function. The sequencer 200 includes sequencer registers 220, an operation store 224 and a sequence controller 228.

A function contains both datapath operations and sequencer operations. Sequencer operations are executed in the sequence controller 228 and include simple operations such as increments, decrements and conditionals. As described above, the datapath operations are executed by the math unit 204.

In order to execute a function, the CPU 104 writes the datapath and sequencer operations to the operation store 224 by way of the sequencer registers 220. The sequencer registers 200 contain other configuration information to assist handshaking between the sequencer 200 and CPU 104. Similar to the configuration registers 208, the sequencer registers 220 appear in the memory map of the CPU 104 and are addressable from the system bus 124 like any other part of the memory map.

The operation store 224 holds the datapath and sequence operations which comprise the function. A memory block within the operation store 224 is used to hold these operations. Decode logic within the operation store 224 assists in reading and writing the memory block. In this embodiment, the memory block is sixty-four words long where each word is thirty-two bits wide. A program counter in the sequence controller 228 is used to sequence through the operations in the operation store.

The sequence controller 228 uses the operations in the operation store 224 along with any information in the sequencer registers to execute the function. Included in the sequence controller 228 are a program counter, operand registers, an arithmetic logic unit (ALU), and a stack. As mentioned above, the program counter addresses the operations in the operand store 224. The stack allows saving the program counter values and any operand registers in order to execute sub-functions which may be part of a function. Increment, decrement and conditional sequencer operations are executed in the ALU so that looping and branching is possible.

When the sequencer 200 encounters a datapath operation, it is sent to the math unit 204 for execution. The sequence controller 228 writes the configuration registers 208 with the necessary information so that the math operator 232 can properly address the operands. Additionally, the datapath operation is written to the configuration

registers 208 so that the math operator 232 can formulate the proper word operations necessary to implement the datapath operation.

As the datapath operations are encountered, the sequencer 200 sends them to the math unit 204 for execution. In contrast, when the sequencer operations are encountered they are executed locally by the sequencer 200. In this way, a function which contains many datapath and sequencer operations can be executed without intervention by the CPU 104 which allows the CPU 104 to attend to other tasks while the crypto engine 108 executes the function. Additionally, system bus 124 bandwidth is preserved because communication from the CPU 104 is not needed while the crypto engine 108 is executing the function. Upon completion of the function, a flag is set for the CPU 104 so that it knows when to retrieve the output operand from the operand memory 216.

With reference to Fig. 3, a flow diagram depicts the process for executing a single datapath operation sent from the CPU 104. As mentioned above, the crypto engine 108 can execute single datapath operations or functions which may contain many datapath operations as well as sequencer operations. Before beginning execution of any datapath operation, the CPU 104 can query the configuration registers 208 to determine if the crypto engine 108 is available for processing the datapath operation. If available, the CPU writes the input operand(s) into operand memory 216 in step 300. As mentioned above, there can be one to three input operands for the various datapath operations. The access to the operand memory 216 is through the math datapath 236 for writing data and through the address generator 240 for addressing data.

In step 304, information is written to the configuration registers 208 by the CPU 104. This information includes operand location, operand size and datapath operation desired. After the configuration information is written, the operation start flag is activated in step 308. In this embodiment, the operation start flag corresponds to a bit in a configuration register which is written by the CPU. However, any method for setting a flag could be used to begin execution of the operation.

At this point in the process, execution within the crypto engine 108 begins. To indicate to the CPU 104 that execution has begun, a busy flag in the configuration registers 208 is set. In step 312, the datapath operation is executed. As discussed above, this involves a state machine in the math operator 232 which issues a series of word operations. The CPU 104 is notified when the execution has completed by setting an execution complete flag in step 316. The complete flag is a bit in the configuration

registers 208. However, other embodiments could use a discrete signal, such as an interrupt, instead of a status bit. In step 320, the output operand is read from the operand memory 216 by the CPU 104 to complete the process of executing a datapath operation.

Referring next to Fig. 4, a process for executing a function is illustrated in block diagram form. In step 400, the CPU 104 writes the function into the operation store 224 within the sequencer 200. The operations which comprise the function are formulated by the firmware running on the CPU 104. Examples of functions which might be sent the crypto engine 108 include RSA key generation, data encryption or data decryption. After the function is stored in the crypto engine 108, the input operands are written into operand memory in step 404. During execution the various datapath operations which comprise the function, these input operands will take many intermediate forms before resulting in an output operand.

Once all the input information is written by the CPU 104, the processing of the function begins. At this point, the CPU 104 is free to execute other unrelated tasks until the crypto engine 108 completes execution. In step 408, a sequencer start flag is activated to signal the crypto engine 108 to begin processing of the input operands according to the function. The first operation in the operation store 224 is fetched in step 410, whereafter the program counter is incremented to point to the next operation in the store 224. In step 412, a determination is made whether the operation is a datapath operation for execution by the math unit 204 or a sequencer operation for execution by the sequencer 200.

If a datapath operation, processing continues to steps 416 and 420. In step 416, the sequencer 200 reads information from the program store 224 and writes the appropriate information into the configuration registers 208. Execution, based upon the information in the configuration registers 208, is performed in step 420. As discussed above, execution involves a state machine within the math operator 232 which manipulates the address generator 240 and math datapath 236 to perform the datapath operation on the input operands in the operand memory 216.

If it is determined in step 412 that the operation is a sequencer operation, processing proceeds to step 424. Execution of the sequencer operation involves performing the operation with the ALU and operand registers within the sequence controller 228. As described above, the sequencer operations are used to branch and loop within the function.

After either executing a datapath or sequencer operation, a determination is made in step 428 whether execution of the function has completed. If there are more operations to perform, processing loops back to step 410 where the next operation is fetched. If execution of the function has completed, the sequencer sets the execution
5 complete flag in the configuration registers 208 in step 430. The execution complete flag signals the CPU 104 that the output operand is read for retrieval. In step 432, the output operand is retrieved by the CPU 104. In this way, a large function with many operations is executed without intervention by the CPU 104.

In light of the above description, a number of advantages of the present
10 invention are readily apparent. The above described crypto engine is faster than software implementations run on general purpose processors. Additionally, the functions are not hard coded into the crypto engine which allows for subsequent changes in the algorithm. Further, the crypto engine requires less circuitry than specialized hardware cryptographic processors because the math operator reuses word size operators in the math datapath.

15 A number of variations and modifications of the invention can also be used. For example, the crypto engine has the ability to execute single datapath operations. In some embodiments, this capability could be removed so that only functions could be executed by the sequencer. However, the functions could only contain one datapath operation. Additionally, some embodiments could use the crypto engine for
20 any variety of cryptographic processing or hash operations using any number of different algorithms. Further, the present invention is not limited to executing a single function at a time. Other embodiments could pass a number of functions to the crypto engine for execution before the CPU retrieves the result.

The forgoing description of the invention has been presented for the
25 purposes of illustration and description and is not intended to limit the invention. Variations and modifications commensurate with the above description, together with the skill or knowledge of the relevant art, are within the scope of the present invention. The embodiments described herein are further intended to explain the best mode known for practicing the invention and to enable those skilled in the art to utilize the invention in
30 such best mode or other embodiments, with the various modifications that may be required by the particular application or use of the invention. It is intended that the appended claims be construed to include alternative embodiments to the extent permitted by the prior art.

WHAT IS CLAIMED IS:

- 1 1. A method for executing a plurality of commands in a cryptographic
2 processing system, the method comprising steps of:
3 receiving a first plurality of commands;
4 executing the first plurality of commands;
5 setting a flag to indicate completion of the step of executing the first
6 plurality of commands;
7 receiving a second plurality of commands; and
8 executing the second plurality of commands.
- 1 2. The method for executing a plurality of commands in a
2 cryptographic processing system as set forth in claim 1, further comprising a step of
3 setting the flag to indicate completion of the step of executing the second plurality of
4 commands.
- 1 3. The method for executing a plurality of commands in a
2 cryptographic processing system as set forth in claim 1, further comprising a step of
3 storing the first plurality of commands.
- 1 4. The method for executing a plurality of commands in a
2 cryptographic processing system as set forth in claim 1, wherein the step of executing the
3 first plurality of commands further comprises a step of converting one command into a
4 plurality of subcommands.
- 1 5. The method for executing a plurality of commands in a
2 cryptographic processing system as set forth in claim 1, wherein the step of setting a flag
3 comprises activating an interrupt.
- 1 6. The method for executing a plurality of commands in a
2 cryptographic processing system as set forth in claim 1, wherein the step of receiving a
3 first plurality of commands comprises receiving a first function.
- 1 7. The method for executing a plurality of commands in a
2 cryptographic processing system as set forth in claim 1, wherein the step of executing a
3 first plurality of commands comprise at least one of branching and looping within the first
4 plurality of commands.

1 8. A cryptographic processing system, comprising:
2 a cryptographic processor which executes a function;
3 a general purpose processor which sends the function to the cryptographic
4 processor; and
5 a bus coupling the general purpose processor to the cryptographic
6 processor.

1 9. The cryptographic processing system as set forth in claim 8,
2 wherein the cryptographic processor comprises a sequencer which stores the function and
3 executes a subset of the commands within the function.

1 10. The cryptographic processing system as set forth in claim 9,
2 wherein the sequencer stores a plurality of functions.

1 11. The cryptographic processing system as set forth in claim 8, further
2 comprising a memory which stores input and output operands for a command.

1 12. The cryptographic processing system as set forth in claim 8, further
2 comprising an interrupt signal which indicates completion of execution by the
3 cryptographic processor.

1 13. The cryptographic processing system as set forth in claim 8,
2 wherein the function comprises a plurality of commands.

1 14. The cryptographic processing system as set forth in claim 8,
2 wherein the cryptographic processor includes a math unit which converts a command into
3 a plurality of subcommands.

1 15. The cryptographic processing system as set forth in claim 8,
2 wherein the cryptographic processor performs loops and branches within the function
3 independently from the general purpose processor.

1 16. A cryptographic processor, comprising:
2 a bus interface;
3 a sequencer which stores a function received from the bus interface;
4 operand memory; and

5 a math unit coupled to the bus interface, operand memory and sequencer.

1 17. The cryptographic processor as set forth in claim 16, wherein the
2 operand memory comprises:

3 a input operand; and

4 a output operand.

1 18. The cryptographic processor as set forth in claim 16, wherein the
2 math unit comprises:

3 a math datapath which executes math operations;

4 an address generator; and

5 a math operator which is coupled to the math datapath and address
6 generator.

1 19. The cryptographic processor as set forth in claim 16, wherein the
2 sequencer executes a subset of the commands within the function.

1 20. The cryptographic processor as set forth in claim 16, wherein the
2 sequencer stores a plurality of functions at one time.

1 21. The cryptographic processor as set forth in claim 16, further
2 comprising a flag which indicates completion of execution by the cryptographic
3 processor.

1 22. The cryptographic processor as set forth in claim 16, wherein the
2 function comprises a plurality of commands.

1 23. The cryptographic processor as set forth in claim 16, wherein the
2 math unit converts a command into a plurality of subcommands.

1 24. The cryptographic processor as set forth in claim 16, wherein the
2 cryptographic processor performs loops and branches within the function independently
3 from a general purpose processor.

1 25. A method for processing cryptographic functions, the method
2 comprising:

3 receiving a function comprised of a plurality of commands which include
4 at least one of a loop and a branch;
5 converting at least one of the plurality of commands to a plurality of
6 subcommands; and
7 executing the plurality of subcommands.

1 26. The method for processing cryptographic functions as set forth in
2 claim 25, further comprising a step of activating a flag in response to the executing step.

1 27. The method for processing cryptographic functions as set forth in
2 claim 25, further comprising a step of storing the first plurality of commands.

1 28. The method for processing cryptographic functions as set forth in
2 claim 25, wherein the converting and executing steps are performed independently of a
3 general purpose processor.

1 29. The method for processing cryptographic functions as set forth in
2 claim 25, further including a step of dividing the plurality of commands between two
3 different processing blocks.

1/4

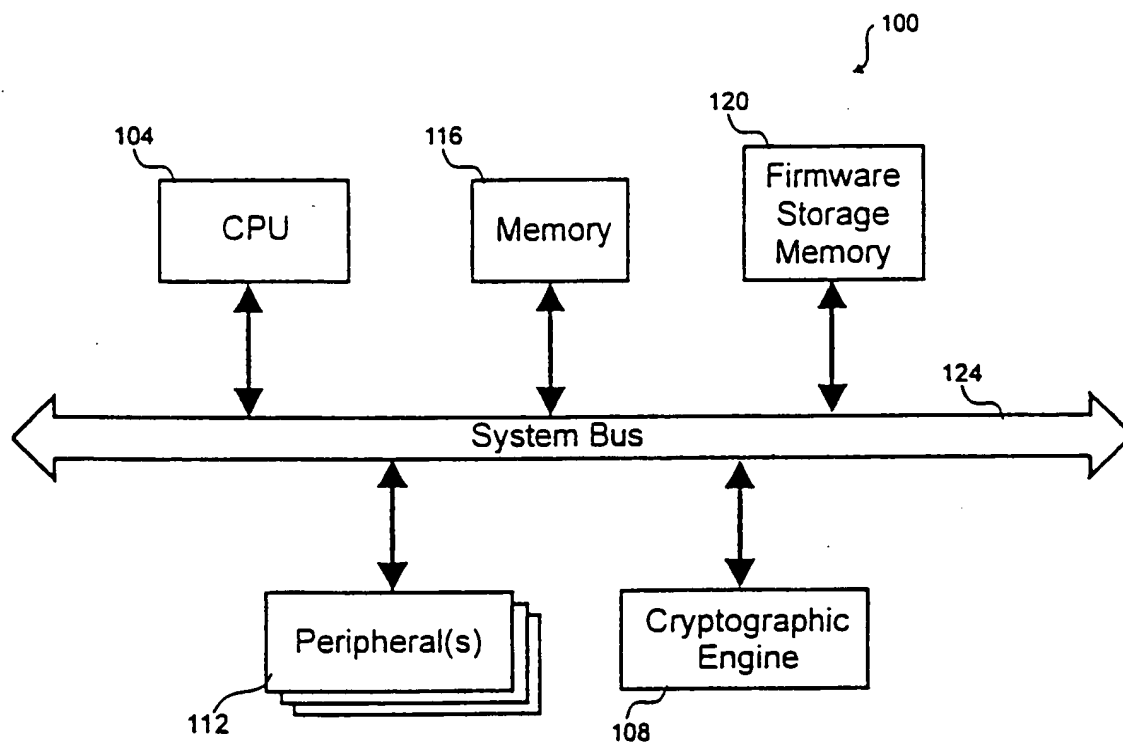


Fig. 1

2/4

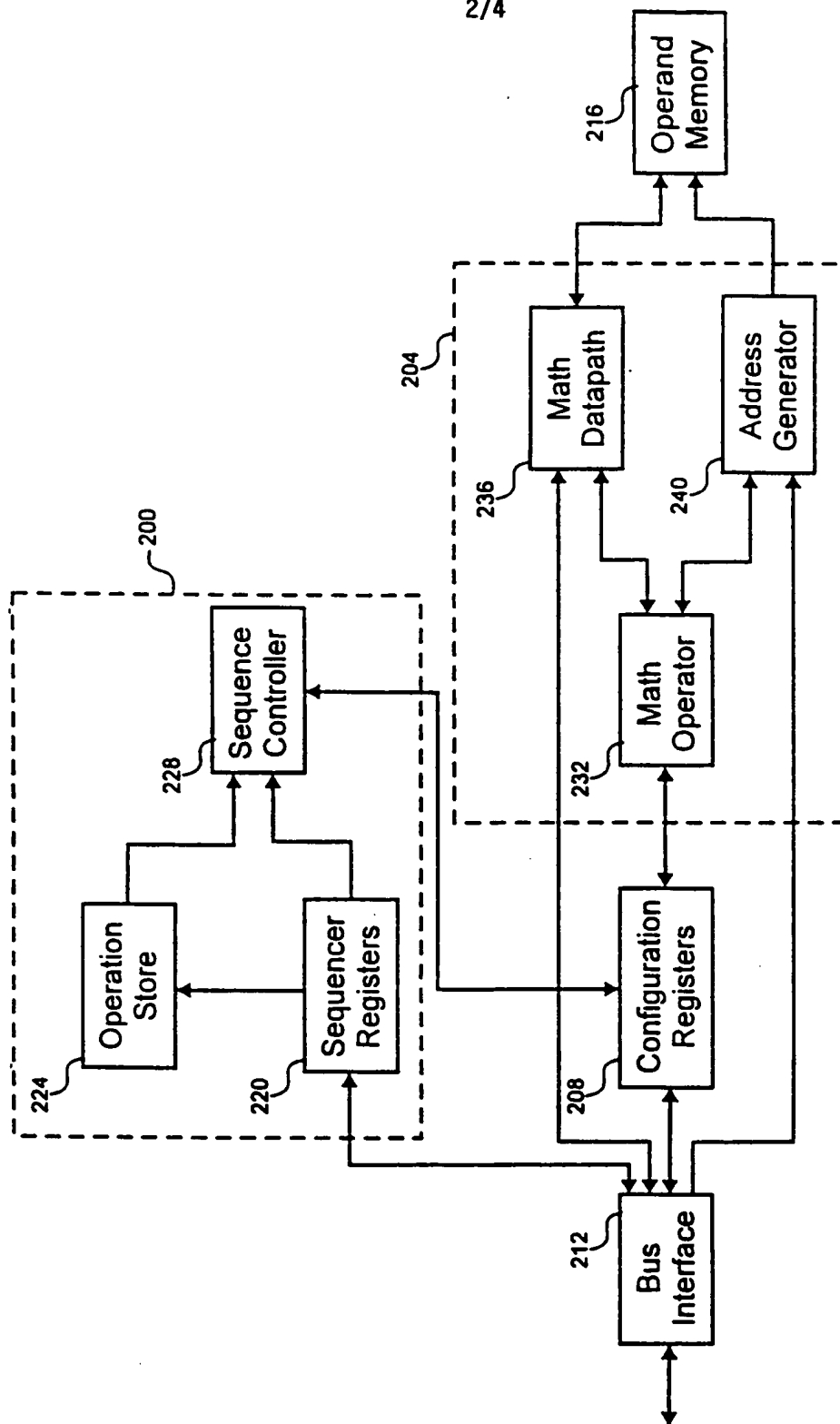


Fig. 2

3/4

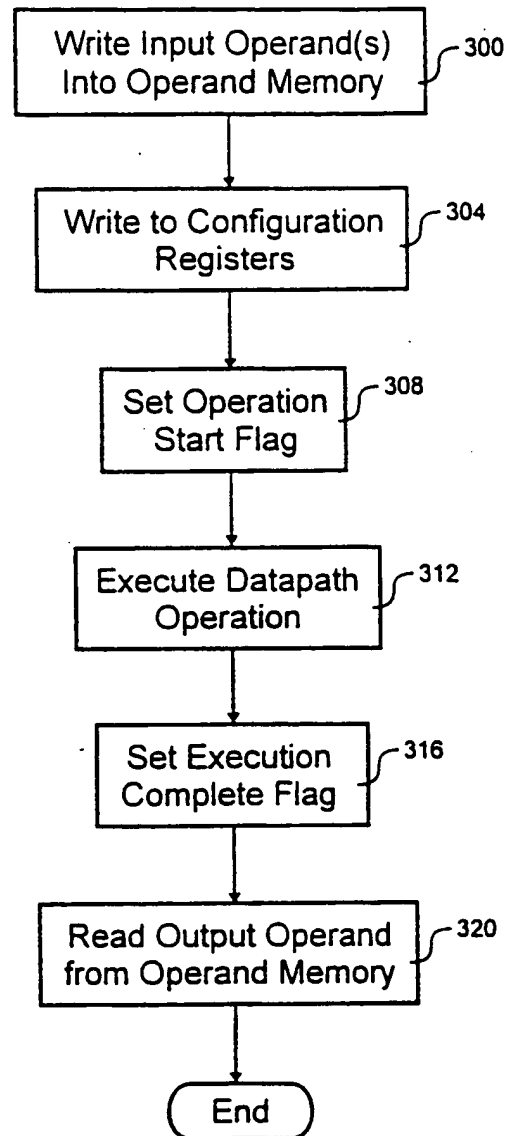


Fig. 3

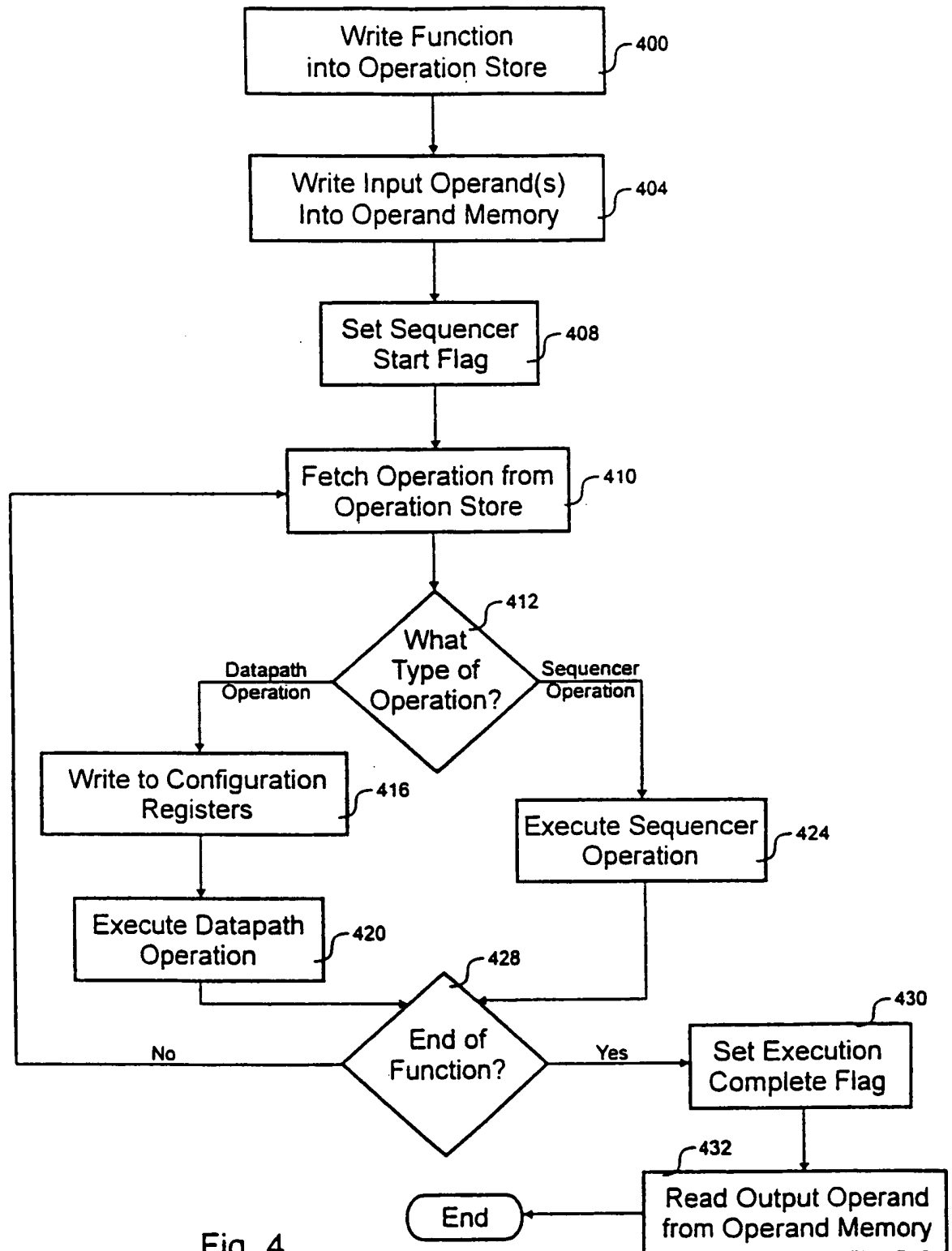


Fig. 4

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US00/15872

A. CLASSIFICATION OF SUBJECT MATTERIPC(7) : H04L 009/32
US CL : 380/28; 713/189, 191

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHEDMinimum documentation searched (classification system followed by classification symbols)
U.S. : 380/28; 713/189, 191

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
BRS (cryptographic near2 coprocessor)**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X — Y	US 5,111,504 A (ESSERMAN) 05 May 1992 (05.05.1992), fig. 1, items 14, 16, 18, 20, 24, 26, 30, 34, 38 and associated texts.	8,11,13 9-10,12,14-15
X	US 5,844,986 A (DAVIS) 01 December 1998 (01.12.1998), fig. 1, items 30, 33, 34 and associated texts.	8
X,P — Y,P	US 6,026,490 A (JOHNS-VANO et al) 15 February 2000 (15.02.2000), column 5, lines 39-50; column 8, lines 9-17; fig. 1, items 200, 202, 300, 301, 500, 501, 502; fig. 3, items 702, 706, 708, 710, 712, 716, 720 and associated texts.	1-7,16-29 9-10,12,14-15

☐ Further documents are listed in the continuation of Box C.☐ See patent family annex.

Special categories of cited documents:	
* "A" document defining the general state of the art which is not considered to be of particular relevance	"T" later documents published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reasons (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"A" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

09 August 2000 (09.08.2000)

Date of mailing of the international search report

25 AUG 2000

Name and mailing address of the ISA/US

Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703)305-3230

Authorized officer

Gail O. Hayes

James R. Matthews

Telephone No. 703 305 3900

Form PCT/ISA/210 (second sheet) (July 1998)

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.